



Network of Excellence  
on Embedded Systems Design

RoSym 2010

## 1st international workshop on Model Based Engineering for Robotics: RoSym'10

October 5th, 2010    Oslo, Norway

The workshop is co-located with **MODELS'2010**-  
supported by  
**Robotics Task Force** at OMG

### Organizers Committee

---

Laurent Rioux, Thales, France  
Sebastien Gerard, CEA-LIST, France  
Davide Brugali, University of Bergamo, Italy

### Program Comittee

---

Xavier Blanc, LIP6, France  
Tetsuo Kotoku, AIST, Japan  
Benoit Potier, Gostai, France  
Nicolas Du Lac, Intempora, France  
Christian Schlegel, University of Ulm, Germany  
Bruno Patin, Dassault Aviation, France  
Cristina Vicente, University of Technology of Cartagena, Spain  
J.F Broenink, University of Twente, Netherland  
Paul Valckenaers, University of Leuven, Belgium  
Toby McClean, ZeligSoft, Canada

## Preface

This volume contains the papers presented at the 1st international workshop on Model Based Engineering for Robotics (RoSym'10), held on October 5th, 2010 in Oslo, Norway in conjunction with the MODELS 2010 Conference.

The main objectives of this workshop are to organize common discussions within Model-based Engineering (MBE) and Robotics experts on how MBE can help robotics people and to share issues that robotics people have encountered with MBE. Current engineering approaches for robotic systems have indeed been demonstrated to be insufficient to bypass following constraints that robotics embedded systems are currently facing:

- the problem space is huge: as uncertainty of the environment and the number and type of resources available to the robot increase, the definition of the best matching between current situation and correct robot resource exploitation becomes overwhelming even for the most skilled robot engineer,
- the solution space is huge: in order to enhance robustness of complex robotic systems, existing cognitive methods and techniques need to exploit robotic-specific resources adequately. This means that the robotic system engineer should master highly heterogeneous technologies in order to integrate them in a consistent and effective way.

One ideal process for developing robotic software components is to enable the design and implementation of highly complex and robust robotic systems to involve in less effort as possible. Robotics systems are complex and embedded ones; thanks to MBE that has already demonstrated its efficiency on complex and embedded systems. We expect MBE to be a real promising solution for the development process of robotics software and systems.

Potentially, new MBE techniques have to be developed for robotics which can also be applicable to other domains. Since robotics is a very challenging domain, we are confident that new techniques may possibly open new way for Model Based Engineering.

September 2010

Laurent Rioux  
Davide Brugali  
Sebastien Gerard

# Conference Organization

## Programme Chairs

- Laurent Rioux
- Davide Brugali
- Sebastien Gerard

## Programme Committee

- Xavier Blanc
- J.F Broenink
- Nicolas DuLac
- Tetsuo Kotoku
- Toby McClean
- Bruno Patin
- Benoit Potier
- Christian Schlegel
- Paul Valckenaers
- Cristina Vicente

## Table of Contents

Using Models@Runtime for Designing Adaptive Robotics Software: an Experience Report . . . . .	1
<i>Juan F. Ingl��s-Romero, Cristina Vicente-Chicote, Brice Morin, Barais Olivier</i>	
Integrating Ontological Domain Knowledge into a Robotic DSL . . . . .	12
<i>Gaelle Lortal, Saadia Dhouib</i>	
Model Driven Embedded Systems Design Environment for the Service Robotics Domain . . . . .	27
<i>Martijn Rooker</i>	
Model-based design of Intelligent Mobile Robot . . . . .	36
<i>Takahiro TAKASU</i>	

# Using Models@Runtime for Designing Adaptive Robotics Software: an Experience Report<sup>\*</sup>

Juan F. Inglés-Romero<sup>1</sup>, Cristina Vicente-Chicote<sup>1</sup>,  
Brice Morin<sup>2</sup>, and Olivier Barais<sup>2</sup>

<sup>1</sup> Dept. of Information and Communication Technologies  
Technical University of Cartagena  
Edif. Antigones (ETSIT), 30202 Cartagena, Spain  
`juanfran.ingles@upct.es`, `cristina.vicente@upct.es`  
<sup>2</sup> INRIA, Centre Rennes - Bretagne Atlantique  
and IRISA, University of Rennes 1  
Campus de Beaulieu, 35042 Rennes Cedex, France  
`Brice.Morin@inria.fr`, `barais@irisa.fr`

**Abstract.** Robotic systems are becoming increasingly complex, as their tasks and working environments become ever richer. As a result, there is an urgent need to provide robots with self-awareness and self-adaptation capabilities that allow them to autonomously deal, among other things, with software and hardware failures, changes in the environment, or interactions with other systems. The use of high-level models that can be adapted at run-time by the robot itself, promises to significantly boost the applicability and performance of robotic systems. This paper reports our experience in applying the DiVA model-driven adaptive approach to a robotics case study, describing its benefits and limitations for robotics.

## 1 Introduction

With increased flexibility and ease of use, robots are at the dawn of a new era, turning them into ubiquitous helpers to improve our quality of life by delivering efficient services in our homes, offices, and public places. In order to achieve such flexibility, the management of uncertainties will be a key component of success [17]. Enabling robots to manage the different sources of uncertainty they must deal with (e.g., changes in the environment, altered requirements, software and hardware failures, etc.) requires providing them with self-awareness and self-adaptation capabilities [2]. This implies enabling robots to build and dynamically adapt models of themselves and their environments.

The Strategic Research Agenda (SRA) [17], delivered one year ago by the European Robotics Technology Platform, defines *adaptation* as *a change to the process or the method of execution performed by the system itself, generally at*

---

<sup>\*</sup> This work has been partially funded by the EXPLORE project (Spanish MICINN, TIN2009-08572, [http://www.dsie.upct.es/proyectos/web\\_explore/](http://www.dsie.upct.es/proyectos/web_explore/)) and the DiVA project (EU FP7 STREP, contract 215412, <http://www.ict-diva.eu/>)

*runtime*. Adaptation may involve cognitive decision making and can take place over both short and long timescales, affecting any level of the system. According to the SRA, future robots, and later groups of robots, will adapt their hardware and software to changes of the environment, work piece, and processes.

Among the mid- and long-term challenges related to adaptation (spanning dimensions such as control, learning, modeling, etc.), the SRA highlights the need for *more automatic (or semi-automatic) use of models for different purposes, including [...] adaptation and reconfiguration*. In this vein, the Model-Driven Engineering (MDE) paradigm promises to bring great benefits to robotics [3].

In a context different from robotics, the DiVA Project<sup>3</sup> proposes to leverage models both at design-time and runtime (*models@runtime*) to support the dynamic adaptation of complex software systems. This paper reports our experience in applying the DiVA approach to a robotics case study, describing its benefits and limitations for robotics.

The rest of the paper is organized as follows: Section 2 surveys related work; Section 3 briefly introduces *models@runtime* in the context of the DiVA Project; Section 4 describes our experience in applying the DiVA model-driven adaptive approach to a robotics case study; Section 5 reports the lessons learned and open challenges; and, Section 6 concludes and presents some future research lines.

## 2 Related Work

Since the late 90s, great research efforts have been made in self-adaptive and autonomic software development. As a result, some interesting high-level reference models and frameworks have been developed [14,9]. In addition, these efforts have also resulted in modern execution platforms, such as Fractal [7], OSGi<sup>4</sup> or SCA<sup>5</sup>, which provide APIs for software introspection and reconfiguration. These platforms currently exhibit some limitations as, for instance, they do not allow to preview the effects of a reconfiguration until it is actually executed, or to simulate *what-if* scenarios in order to evaluate different possible configurations *a priori*. Moreover, in the case of complex adaptive systems, a large number of low level reconfiguration scripts (calls to the reconfiguration API) need to be manually coded, making the process cumbersome and error prone.

Putting the focus on the robotics domain, some interesting results have been achieved by the bio-inspired and cognitive system communities on low-level robot behavior adaptations based, e.g., on genetic algorithm mutations [10]. However, in order to deal with the increasingly growing complexity of real-world robotic systems and working environments, higher-level adaptation mechanisms need to be developed. In this vein, it becomes necessary to shift the focus from low-level self-adaptive algorithms to higher-level self-adaptive software components and component-based architectures [9]. Furthermore, the envisaged adoption of MDE by the robotics community promises not only to help raising the level of

---

<sup>3</sup> <http://www.ict-diva.eu/>

<sup>4</sup> <http://www.osgi.org>

<sup>5</sup> <http://www.eclipse.org/stp/sca/>

abstraction at which robotics systems are designed, but also enable their self-adaptation using models@runtime.

Although there exist plenty of robotics-specific software architecture styles and frameworks, commonly supported by platform-specific (and hardly interoperable) middlewares [15], most of them currently lack of support for model-driven robotics software development and self-adaptation [11]. Among the few model-driven tool-chains for robotics software development, it is worth highlighting Smartsoft [18] and V3CMM [6], both enabling component-based platform-independent design modelling and platform-specific code generation by means of model transformations. However, to date, neither Smartsoft nor V3CMM support runtime software adaptation (V3CMM only supports structural and behavioural variability modelling at design-time). Conversely, the work presented in [8], addresses robotics software runtime adaptation at an architectural level although, having not adopted a MDE approach, it strongly depends on the Prism-MW<sup>6</sup> specific middleware platform.

Finally, it is worth highlighting the very interesting initiative started by the BRICS project<sup>7</sup>, funded by the 7<sup>th</sup> EU Framework Program, where both MDE and robotic system adaptation (to achieve robust autonomy) play a key role, although these two goals do not appear explicitly related in the proposal.

### 3 An Overview of DiVA

The idea of “models@runtime” is to leverage models both at design-time and runtime to monitor, dynamically adapt or evolve software systems. A dedicated workshop<sup>8</sup> is held at MODELS since 2006.

In the context of the DiVA project [16,5], we leverage models@runtime to support the design and the execution of Dynamic Software Product Lines (DSPL) [12]. At design-time, we describe four facets of a DSPL, that are then leveraged at runtime to drive the dynamic adaptation process:

- **Variability**: describes the different features of the system, and their natures (options, alternatives, etc)
- **Environment/Context**: describes the relevant aspects of the context we want to monitor (environment), as well as the current context.
- **Reasoning**: describes when the system should adapt. It consists in defining which features (from the variability model) to select, depending on the current context, using the appropriate formalisms.
- **Architecture**: describes the configuration of the running system in terms of architectural concepts.

The role of these models is to formalize how and when a system should adapt. Thus, adaptation models capture the variability in the system and in its context, and link changes in the latter with configurations of the former.

---

<sup>6</sup> <http://csse.usc.edu/~softarch/Prism/>

<sup>7</sup> <http://www.best-of-robotics.org/>

<sup>8</sup> <http://www.comp.lancs.ac.uk/~bencomo/MRT/>

It is important to note that designers do not specify the whole possible set of architecture configurations in extension. Instead, each feature of the variability model is refined into an aspect model that can be easily woven into a base model (which contains components and bindings that should be present in all configurations). This way, the system is designed in intention, and configurations are explicitly built when needed. When a configuration is built (by aspect weaving) it is first validated by checking some invariants. Then if the configuration is valid, we rely on a model comparison (between the current configuration and the newly produced one) to infer a safe migration path that is actually executed to adapt the running system. This prevents designers from writing low-level and error-prone reconfiguration scripts. Interested readers are referred to [16,5] for more details about the use Aspect-Oriented Modeling in DiVA.

## 4 Applying Models@Runtime to a Robotics Case Study

### 4.1 Case Study Description

To illustrate how DiVA can address adaptation in robotics, we present a simple case study developed using its current version. This experience will allow us to later discuss the advantages and drawbacks of DiVA in the context of robotics.

The case study takes place in a room, containing a number of obstacles, where two commercial robots (e-pucks<sup>9</sup>) are initially placed at arbitrary positions. One of them plays the role of *Victim*, while the other plays the role of *Rescuer*.

The goal of the *Victim* is to help the *Rescuer* find it as soon as possible. To achieve this, it indicates its position using an acoustic or light signal. The *Victim* uses the Bluetooth to communicate both changes in its signaling policy and its current state, which can be: *i*) OK, *ii*) Wounded, or *iii*) KO. The *Victim* is equipped with infrared (IR) proximity sensors to detect close objects, which it can avoid adopting different strategies, namely: *i*) surround the obstacle, or *ii*) change the movement direction. The first action has a greater impact on energy consumption. Additionally, the *Victim* can adopt different strategies to improve its visibility, namely: *i*) run randomly, *ii*) walk randomly, or *iii*) stay still, in descending order in terms of visibility and resource consumption.

The goal of the *Rescuer* is to find the *Victim* in the shortest time possible with the available resources. It is equipped with three sensors for this purpose, each one having a different precision and consumption: *i*) camera, *ii*) microphones (which can identify the direction of sound), and *iii*) IR proximity sensors. It can receive Bluetooth communications from the *Victim*, allowing it to select the most appropriate sensor and strategy to find it. The *Rescuer* uses the same obstacle avoidance strategies as the *Victim*. Both robots are equipped with sensors to measure environmental light and noise, and their battery level.

Robots are expected to dynamically adapt their behaviour depending on their context (i.e., their role, battery level, light conditions, etc.) in order to achieve their goals using the most appropriate sensors and strategies.

---

<sup>9</sup> <http://www.e-puck.org>



## 4.2 Modeling Dynamic Variability and Adaptation

As described in Section 3, DiVA considers four facets of a DSPL: *Variability*, *Environment/Context*, *Reasoning*, and *Architecture*. Next, we present the adaptation models developed for the case study using the DiVA Eclipse-based editors.

Fig. 1 shows how the context of both robots is modeled. In both cases, three boolean context variables capture the changes in the environmental light and noise, and in the robot battery. The *Rescuer* includes two additional context variables: *Signal Notification* and *Victim State*, which capture the changes in the *Victim*'s signaling policy and state. Note that, whenever the *Rescuer* has no information about the *Victim*, these variables are set to the UNKNOWN value. For the *Victim*, only one additional variable is modelled: *State*, which is set by one of the *Victim*'s internal components according, e.g., to the time it has been lost.

In robotics, we can think of three sources of contextual information having impact in robot software adaptation: *i*) the environment, *ii*) the robot internal state and resources, and *iii*) the perception of (and, eventually, the communication and collaboration with) other systems, either robotics or not.

Fig. 2 shows the variability model including the dependencies among the variants and the adaptation constraints. The variants represent different possible realizations of a variability dimension. For instance, there are three variants for the *Search Strategy* dimension, one for each of the strategies the *Rescuer* can use to find the *Victim*. As the cardinality of this dimension is set to [1..1], one (and only one) of the strategies needs to be selected for each particular configuration of the *Rescuer*. The *Detailed* strategy involves the highest search accuracy, and thus requires the camera and microphones variants (see Dependency column). The Available and Required expressions correspond to contexts in which the variant respectively can or must be used. For example, given the importance of energy consumption, it only makes sense to consider the *Detailed* strategy when the battery of the robot is not low.

The next step is to model the properties relevant for the system, i.e., the functional and extra-functional properties that need to be optimized. Each property has a name and a direction, the later specifying if it should be minimized (0)













	Name	ID
Enum	Signal Notification	Signal
Literal	LIGHT	LIGHT
Literal	ACUSTIC	ACUSTIC
Literal	Unknown	USIG
Enum	Victim State	STATE
Literal	Unknown	UNKN
Literal	OK	OK
Literal	Wounded	Wounded
Literal	KO	KO
Boolean	Low Battery	LowBatt
Boolean	Ambient Light	LIGHT
Boolean	Ambient Noise	NOISE

(a)













	Name	ID
Boolean	Low Battery	LowBatt
Boolean	Ambient Light	LIGHT
Boolean	Ambient Noise	NOISE
Enum	State	STATE
Literal	OK	OK
Literal	Wounded	WO
Literal	Cannot Move	KO

(b)

Fig. 1. Context model for the robot playing the role of (a) *Rescuer* and (b) *Victim*.



	Name	ID	Lower	Upper	dependency	available	required
 Dimension	Sensors	SEN	0	2	-	-	-
 Variant	Camera	CAM	-	-	not BS	LIGHT or Signal=LIGHT	-
 Variant	Microphones	MIC	-	-	not BS	not NOISE or Signal=ACUSTIC	-
 Dimension	Networking	NET	0	1	-	-	-
 Variant	Bluetooth	BT	-	-	-	-	not LowBatt
 Dimension	Search Strategy	SST	1	1	-	-	-
 Variant	Detailed	DS	-	-	CAM and MIC	not LowBatt	-
 Variant	Simple	SS	-	-	CAM or MIC	-	-
 Variant	Blind	BS	-	-	-	-	LowBatt
 Dimension	Obstacles Strategy	OBSST	1	1	-	-	-
 Variant	Surround	SUR	-	-	-	-	-
 Variant	Change Direction	CD	-	-	-	-	-

(a)



	Name	ID	Lower	Upper	dependency	available	required
 Dimension	Networking	NET	0	1	-	-	-
 Variant	Bluetooth	BT	-	-	-	-	not LowBatt
 Dimension	Signaling	SIG	1	1	-	-	-
 Variant	Light Generator	LIGEN	-	-	-	-	-
 Variant	Acoustic Generator	ACGEN	-	-	-	not NOISE	-
 Dimension	Movement Strategy	MOVST	1	1	-	-	-
 Variant	Random running	RUN	-	-	SUR or CD	STATE=OK	-
 Variant	Random walk	WALK	-	-	SUR or CD	not STATE=KO	-
 Variant	Stay Still	STAY	-	-	-	not STATE=OK	-
 Dimension	Obstacles Strategy	OBSST	0	1	-	-	-
 Variant	Surround	SUR	-	-	not(STAY)	-	-
 Variant	Change Direction	CD	-	-	not(STAY)	-	-

(b)

**Fig. 2.** Model of the variability and constraints for (a) *Rescuer* and (b) *Victim*.

	Name	Direction
 Property	Power consumption	0
 Property	Search accuracy	1

(a)

	Name	Direction
 Property	Power consumption	0
 Property	Signaling accuracy	1

(b)

**Fig. 3.** Selected properties for (a) *Rescuer* and (b) *Victim*.

or maximized (1). As shown in Fig. 3, we have selected to minimize the power consumption, and maximize the search and signaling accuracy.

Fig. 4 shows the impact of variants (rows) on each property (columns). When a dimension has an impact on a certain property, for each of its variant a qualitative appreciation of this impact has to be specified. For example, the *Signaling* dimension affects both the power consumption and the signaling accuracy. In particular, the *Light Generator* has a low power consumption and signaling accuracy, while the *Acoustic Generator* has a medium power consumption and a high accuracy. This table is the base to make different trade-offs among the variants and to select the optimal configuration for the actual context.

Finally, Fig. 5 shows the robot adaptation rules. These are Priority Rules, i.e., they capture the relevant system properties depending on the context. For example, the rule *Battery is low* specifies that when the battery is low, optimiz-

	Power...	Search...		Power...	Signaling...
Sensors (SEN)	true	false	Networking (NET)	true	true
Camera (CAM)	High	-	Bluetooth (BT)	High	Medium
Microphones (MIC)	Medium	-	Signaling (SIG)	true	true
Networking (NET)	true	true	Light Generator (LIGEN)	Low	Low
Bluetooth (BT)	High	Low	Acoustic Generator (ACGEN)	Medium	High
Search Strategy (SST)	false	true	Movement Strategy (MOVST)	true	true
Detailed (DS)	-	Very High	Random running (RUN)	Medium	Very high
Simple (SS)	-	Medium	Random walk (WALK)	Low	Medium
Blind (BS)	-	Very Low	Stay Still (STAY)	Non-eff...	Non-effect
Obstacles Strategy (OBSST)	true	true	Obstacles Strategy (OBSST)	true	false
Surround (SUR)	Low	Medium	Surround (SUR)	Low	-
Change Direction (CD)	Very Low	Low	Change Direction (CD)	Very low	-

(a)
(b)

**Fig. 4.** Impact of variants on robot properties for (a) *Rescuer* and (b) *Victim*.

Name	context	Power...	Search...	Name	context	Power...	Signaling...
Battery is Low	LowBatt	High	-	Battery is Low	LowBatt	High	-
Unknown Signaling	Signal=USIG	Low	High	Battery is OK	not (LowBatt)	Low	Medium
Unknown State	STATE=UNKN	Low	High	Wounded	STATE = WO	Very Low	Medium
Battery is OK	not LowBatt	Low	-	Almost Dead	STATE = KO	-	Very High
Night	not (LIGHT)	-	Very High	OK	STATE = OK	Medium	Low
Victim OK	STATE=OK	Medium	Medium				
Victim NOK	not STATE=OK	-	Very High				

(a)
(b)

**Fig. 5.** Robot adaptation rules for (a) *Rescuer* and (b) *Victim*.

ing power consumption has a high priority. Conversely, the *Battery is ok* rule specifies that this is a secondary concern when the battery is OK.

It is worth noting that DiVA allows the simulation of the previous adaptation models provided the user inputs a sequence of contexts (set of values for the context variables). For each context, the simulator calculates and shows (it must be said that not in a very friendly and readable way) the best possible architecture configuration. This facility, has allowed us to perform multiple tests on the models developed as part this work, although we decided not include the screenshots showing the results for the sake of simplicity and for the space limitations.

### 4.3 Runtime Architecture to Support Dynamic Variability

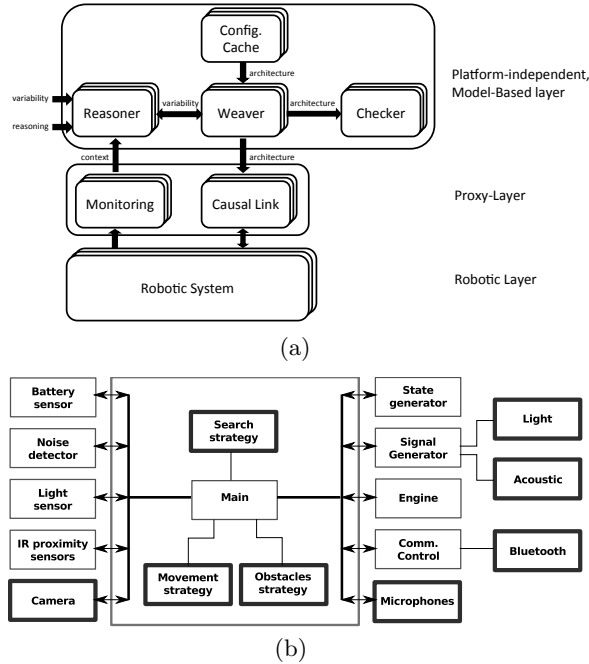
In order to support actual runtime adaptation in robotics, we can consider the three-layer architecture developed as part of the DiVA project (see Fig. 6(a)). In the platform-independent model-based layer, the components produce and consume models, i.e., when the context model is updated, the *Reasoner* calculates a derived variability model accordingly. For all the features included in the variability model, the *Weaver* composes the corresponding architectural model, which then is checked and submitted to the proxy layer. The Proxy layer is responsible for bridging the gap between design-time models and the runtime. The

Causal Link component receives the architectural model and reconfigures the architecture being executed by the robot. Additionally, the *Monitoring* component observes runtime events generated at runtime by the probes in order to create and update the context model. Finally, the Robotic layer basically contains all the specific components of the robot (e.g. see Fig. 6(b)).

## 5 New Challenges for Runtime Adaptation in Robotics

Models@Runtime have demonstrated promising results for dynamically adapting business system architectures. However, the experience reported in this paper shows that several issues remain in this approach when applied to robotics. In this section, we present a set of problems and challenges that we feel that are of particular importance and that need to be addressed in the next stage of Model-Driven Engineering for robotics. These challenges are the result of the lessons learned from the experience described in Section 4.

- *Cooperative adaptation engines.* In DiVA, as in most adaptation approaches, it is not easy to design several adaptation engines that need to cooperate. In robotics, this is a fundamental requirement as robots often need to cooperate with other systems (e.g., other robots). Thus, designers should be able to model all adaptation engines and their relations.



**Fig. 6.** (a) Runtime architecture to support dynamic variability in robotics. (b) Case study robot component. Thick lines denote some variability degree.

- *Multi-layer adaptation engines.* Robots might belong to teams and, eventually, to other higher-order communities. Thus, they might need to adapt themselves according to both individual and global adaptation strategies. The current versions of DiVA can not manage layered engines in which the adaptation can be driven both locally and a globally.
- *Model the impact of context dimensions during simulation.* To obtain a correct simulation, it would be interesting to model the impact of selecting certain context dimensions. For instance, if the Bluetooth variant is deactivated, it should not be possible to update the context (e.g., the state of other robots). However, during the simulation step provided by DiVA, all the context variables can be modified at any time, even if the variant controlling the update of these variables is deactivated. This is a major drawback for simulation. In fact, designers need to modify the context model themselves to obtain a correct simulation at each adaptation step.
- *Context sharing.* DiVA allows the description of the relevant aspects to be monitored (environment) as well as the current context. However, it does not support context sharing, sometimes necessary in robotics (e.g., to allow cooperating robots to share their local maps to obtain a global one).
- *Context uncertainty management.* If we add the possibility of sharing context models, we also need to manage the confidence of the shared information. Thus, it is important to know how safe or accurate is the information shared by each robot, as it can (willingly or not) provide others with useless or even dangerous information. Coupling context information with fuzzy logic could simplify the design of context models [4].
- *Using Model@runtime for implementing not only the system architecture but also its components.* Current approaches, such as DiVA, mainly model the system architecture. As a consequence, the adaptation engine can only work at that level (adding or removing components or bindings among them, changing the value of component attributes, etc.) To support other kinds of adaptation, it can be interesting to use models@runtime for component implementation [13]. This would allow us to also adapt some parts of the component implementation.

These six challenges are not exhaustive. They just aim to describe some of the new requirements that need to be managed to make models@runtime usable in the context of adaptive robotic systems.

## 6 Conclusions and Future Work

This paper reports our experience of using the DiVA model-driven approach to design and implement an adaptive robotic system. Both the benefits and the drawbacks of such an approach have been described. This paper makes the following claims that, in our opinion, are worth being discussed at the workshop:

- As a community, we need to take the next step and adopt the perspective that robotics systems are software intensive systems and their architecture has to

be properly modeled. As stated in [1] software architecture is, *fundamentally, a composition of architectural design decisions* (dimensions in DiVA). These design decisions (that fix some variation points) should be represented as first-class entities in the software architecture and it should be possible to add, remove and change architectural design decisions against limited effort. For that, a “models@runtime” approach like DiVA provides a first answer.

- Models should be used both at design- and at run-time. There is a clear benefit of modeling adaptive robotic systems, as the same models can be used to simulate the robot behavior in a particular context, and then directly executed by the robot at run-time.
- Robotic systems are adaptive systems. They should be resource- and context-aware. As systems with limited resources, robots cannot always be confident in the context and cannot always collect the same level of information.
- Robotic systems should be designed to support cooperation with other systems. In that direction, model-driven engineering for robotics research should share some knowledge and design principles with model-driven engineering for Systems of Systems.

For the future, we plan to address some of the challenges identified in Section 5. In particular, in the short- and mid-term, we will work in two main directions: extending the DiVA framework so it can manage several adaptation engines, and supporting a better context representation.

## References

1. J. Bosch. Software architecture: The next step. In Flávio Oquendo, Brian Warboys, and Ronald Morrison, editors, *EWSA*, volume 3047 of *Lecture Notes in Computer Science*, pages 194–199. Springer, 2004.
2. D. Brugali and A. Shakhimardanov. Component-Based Robotic Engineering. Part II: Models and Systems. *IEEE Robotics and Automation Magazine*, 17(1):100–112, 2010.
3. H. Bruyninckx. Robotics software: the future should be open. *IEEE Robotics and Automation Magazine*, 15(1):9–11, 2008.
4. F. Chauvel, O. Barais, I. Borne, and J.M. Jézéquel. Composition of qualitative adaptation policies. In *23rd IEEE/ACM International Conference on Automated Software Engineering - ASE’08*, L’Aquila, Italy, sep 2008. Short paper.
5. B. Morin et al. Models@Run.time to Support Dynamic Adaptation. *IEEE Computer*, 42(10):44–51, 2009.
6. D. Alonso et al. V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development. *Journal of Software Engineering for Robotics*, 1(1):3–17, 2010.
7. E. Bruneton et al. The FRACTAL Component Model and its Support in Java. *Software Practice and Experience, Special Issue on Experiences with Auto-adaptive and Reconfigurable Systems*, 36(11-12):1257–1284, 2006.
8. G. Edwards et al. Architecture-driven self-adaptation and self-management in robotics systems. *International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 142–151, 2009.

9. P. Oreizy et al. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, 1999.
10. D. Floreano and L. Keller. Evolution of Adaptive Behaviour in Robots by Means of Darwinian Selection. *PLoS Biol*, 8:e1000292, 2010.
11. J. Georgas and R. Taylor. Policy-based self-adaptive architectures: a feasibility study in the robotics domain. In *SEAMS 2008*, pages 105–112, 2008.
12. S. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid. Dynamic Software Product Lines. *IEEE Computer*, 41(4), April 2008.
13. E. Hoefig, P.H. Deussen, and H. Coskun. Statechart Interpretation on Resource Constrained Platforms: a Performance Analysis. In *4th International Workshop on Models@Run.Time (at MODELS'09)*, Denver, Colorado, USA, Oct 2009.
14. J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In *Workshop on the Future of Software Engineering (FOSE 2007)*, pages 259–268.
15. N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Middleware for robotics: A survey. In *IEEE International Conference on Robotics, Automation, and Mechatronics (RAM 2008)*, pages 736–742, 2008.
16. B. Morin, O. Barais, G. Nain, and J. M. Jézéquel. Taming Dynamically Adaptive Systems with Models and Aspects. In *ICSE'09: 31st International Conference on Software Engineering*, Vancouver, Canada, May 2009.
17. European Robotics Technology Platform(EUROP). Robotic visions to 2020 and beyond - the strategic research agenda for robotics in europe. appendix: Technology roadmaps. <http://www.robotics-platform.eu/sra>, 2009.
18. C. Schlegel. Communication patterns as key towards component interoperability. In *Software Engineering for Experimental Robotics*, volume 30, pages 183–210. Springer-Verlag, Berlin, Heidelberg, 2007.

## Integrating Ontological Domain Knowledge into a Robotic DSL

G  lle Lortal, Saadia Dhouib

Decision Technologies and Mathematics Lab., Thales Research & Technology, Campus  
Polytechnique, 1, avenue Augustin Fresnel 91767 Palaiseau cedex France

`gaelle.lortal@thalesgroup.com`

CEA LIST, Laboratoire d'Ing  nierie dirig  e par les mod  les pour les Syst  mes Embarqu  s  
(LISE), Point Courrier 94, Gif-sur-Yvette, F-91191 France

`saadia.dhouib@cea.fr`

**Abstract.** Coming from the Artificial Intelligence (AI) and Semantic Web (SW) circles, ontologies are used mainly to represent domains. The Model Driven Engineering (MDE) field gave birth to Domain Specific Languages to represent a particular technical domain. Abstracting from their uses, we consider as many others researchers that ontologies and models are closer than their original fields could get to think. Furthermore, their building or development are facing the same problems. They are costly and need experts' interviews in order to grasp specific knowledge and structure it. Likewise, ontologies and DSL can benefit from each other domains in reusing construction methodologies and even reusing knowledge modelled in another format. In this paper we first present the ontologies and DSL definition we use and some methodologies of development enabling the reuse of knowledge (as alignment, fusion). We then present how we propose to reuse the knowledge of a robotic ontology to develop robotic DSLs within the PROTEUS<sup>1</sup> project in order to inject ready-made domain information to the DSL.

## 1 Introduction

Following (Caplat, 2008), (Guizzardi, 2007), and (Gasevic, 2005), we consider ontologies and models/metamodels as highly close. To use both technologies with the best interests and find the connection points, we compare them on several criteria. The key connection points to use ontologies and models are about the abstraction levels that they represent (and then the necessary abstraction level to bind them) and more the applications and tools that are available for both technologies (and then check which are reusable or interchangeable).

Observing DSLs and ontologies, the first noteworthy remark is on the building/development methodology that is identical. An ontology as a DSL can be built/developed through inquiry, domain survey and modelling. Then we postulate that for the same objective, we should find the same concepts represented (with different formats) and then that we can find equivalent items from the DSL to the ontology.

---

<sup>1</sup> The PROTEUS project is a three-years funded by the French National Research Agency



In order to check such hypothesis, we built in parallel, from the same use-cases and experts interviews robotic ontology and robotic DSL in the scope of the PROTEUS ANR Project. One of the DSLs is built from the PROTEUS ontology knowledge and the others from scratch. Here is only presented the development methodology of a Robotic Architecture DSL whose requirements are ontology based.

In this paper, we first present a definition of ontology and DSL and their application. In section 3, we then present our analysis grid and in the last section, we present how we propose to reuse the knowledge of a robotic ontology to develop robotic DSLs within the PROTEUS project in order to inject ready-made domain information to the DSL.

## **2 Ontology, DSL: some definitions**

We first present a short state of the Art on ontologies and DSLs within the view adopted in the PROTEUS project. In the first part, we describe ontology as a structure of the data and in the second part, we describe the DSL (Domain Specific Language as a language designed for, and intended to be useful for, a specific kind of concern.

### **2.1 Ontology**

The ontology is one of the favourite tools of the Semantic Web (SW). The SW proposes different tools using normalized data or which helps structuring Web data and associating “semantics” to data. A syntactic layer is added to the data available on the Web and is claimed to be the semantic enrichment. It’s this layer which aims at enabling a mutual machine-machine or man-machine understanding.

Ontology is defined by (Gruber, 1993) as an explicit specification of a conceptualization. In (Gruber and Lytras, 2004), Thomas Gruber refines its definition of this type of Knowledge Base (KB) taking into account the necessary cooperation of experts of the domain to come to an agreement on the semantics, the ontological commitment: “*Every ontology is a treaty – a social agreement – among people with some common motive in sharing*” (p. 5). The process of negotiation is oriented toward the objective of the conceptualization more than towards its structure. T. Gruber strongly emphasizes the idea of a viewpoint carried by the ontology:

*“The ontology is a representation artefact (a specification), distinct from the world it models, and that it is a designed artefact, built for a purpose. [...] I would try to emphasize that we design ontologies.”* (Gruber et Lytras, 2004, p. 1)

(Lassila and McGuinness, 2001) identify more or less formal ontologies and (Uschold and Grintinger, 1996) organize them according to their uses:

- For human communication (not ambiguous ontology but informal);
- For computer systems interoperability (exchange format);
- For system design (formal encoding and metadata).

We can define a formal ontology as a modelling of knowledge of the World. The knowledge is organized on a network of concepts. An ontology, then, consist in a set

of definitions of basic categories (things, relations, properties) which enables to describe the things of the domain of interest, their properties and the relations the things maintain among each others. Then, hierarchical relations (*isa* relations) or horizontal relations among concepts or instances are rigorously defined. The concept properties can have values in finite and predefined intervals and the strictly defined axioms impose logical constraints enabling the control of logical inferences applicable on data (properties inheritance, transitive or inverse properties...). Via these inferences, new knowledge can be discovered. Despite this mechanism, domain experts and knowledge engineers should be involved in the ontology building.

Ontologies are used in several domains. In SW, each element is tagged. The tag is understood by software systems which enables their interoperability (as for Web Services). These tags, normalized and understandable by Human and Software agents, give semantics to the Web. In Artificial Intelligence (AI), the ontologies can be used to mime human behaviours, as for example, human language with linguistic ontologies. In system design (Architecture, Engineering) the ontologies used are formal and propose a complex and rigid modelling of knowledge usable by software agents.

The challenges on ontologies are improvements on knowledge sharing on the Web, systems interoperability, Man-Machine/Machine-machine communication and then step in ontology building (Buitelaar *et al.*, 2005) and update (Cimiano and Völker, 2005), use and reuse of ontologies (annotation (Handschuh and Staab, 2003), fusion/merging/alignment (Noy and Musen, 1999)) and ontology language development (RDF (Brickley *et al.*, 2004), OWL (McGuinness *et al.*, 2004), Topic Maps (Biezunski *et al.*, 1999) etc.).

T. Gruber considers that ontologies are always mixes of informal and formal parts. The ontologies that are said to be semi-formal are mainly informal and he finally concludes that “*all practical ontologies are semiformal*”.

## 2.2 Domain Specific Language

According to (Bezivin, 2004), initially, the objects technologies were supposed to be an integration technology as it was theoretically possible to take into account homogeneously, processes, rules, functions, etc., through objects. Nowadays, we return to less hegemonic vision where the different programming and management paradigms coexist and models are no more considered only as documentary or guiding means for a human activity of programming, but that they can be used to feed tools for software automatic production. The MDE is an integrated vision through DSL based on different paradigms. Indeed, Metamodels and Models are used to ease the whole software lifecycle management, i.e. the code generation but also, integration and interoperability, documentation generation and the automation of software applications deployment. The different levels of Models are represented by M0 - data level/instances, M1 - model level, M2 - metamodel level, M3 - meta-meta-level.

A Domain Specific Language is a formal language, and then a grammar, tailored to a specific application domain. It is then a metamodel at a notation level. Constructs

and abstractions of the domain are offered within the language increasing its expressiveness in comparison to General Purpose Languages (GPL). A DSL (or its graphical representation, the DSML - Domain Specific Modelling Language -) is a textual representation of a domain and enables the specification of a M1 - model in accordance with a defined metamodel (M2). The DSL enables to build and read a model. It adds symbols, represents the concepts of a metamodel and enables to handle them.

A model is an abstraction of the reality (as a Data-Base) and therefore is only a specific viewpoint on the reality.

The metamodel (MM) is a self defining model and also underlies model(s) (whether explicitly or not). The MM establishes the concepts which are useful in a specific Domain of interest and the rules of use together. I.e. it defines the relations among the concepts in a distorted view of the situation which is to say according to a certain viewpoint of the domain.

A large collection of M2 standard metamodels exists to represent specific domains (as well as UML profiles). They are sometimes coupled with specific tools enabling to tackle specific principles or methodology. The OMG proposes the MOF meta-meta language on which generic meta-languages (UML) as well as specific meta-languages are built (OMG, 2006). MOF enables to develop Domain languages (DSL) and UML enables to develop Profiles (as MARTE for Modelling and Analysis of Real-Time and Embedded Systems). Other metamodel implementations are available for BPMN (Business Process Model and Notation) or Information Management Metamodel (IMM) systems (SysML) for example<sup>2</sup>.

The semantics of the Models is not straightforwardly available. It is more inherent to the use of the model (in reading it, transforming it, edit it, modify it...). I.e. the semantics are in the interpretation of a model and the rules applied to transform it (any rules of decoding/recoding). The rules enable to enrich, filter, add, specialize (and even «retro-engineered») information of the model to generate another model. A semi-automated way for interpreting models is available through transformation languages as QVT (Query, Views, Transformation, (QVT, 2008)) or ATL (ATL, 2010) (Lemesle, 1998, and Bezivin, 2004). However, some constraint expression languages express semantics as they constrain the interpretation of the information. It is the case for the OMG standard OCL (Object Constraints Language) or through a verification tool as PRAXIS (Blanc, 2008).

Even if DSL has for main requirement the proven consistency of the model, it appears that the same fundamental questions are posed in the two domains. What to model, how to model it, how to reconcile user needs and system requirements? But also, what does semantics means? What is granularity and what is viewpoint? How concepts evolve? Then we pose a number of criteria to compare ontologies and DSL. We present them on the next section.

---

<sup>2</sup> The Catalog of OMG Modelling and Metadata Specifications is available at: [http://www.omg.org/technology/documents/modeling\\_spec\\_catalog.htm](http://www.omg.org/technology/documents/modeling_spec_catalog.htm)

### 3 Ontology/DSL Comparison

In order to fix the gain of using an ontology in comparison to building a DSL, we ordered our observation on a comparison grid between ontology and DSL. We present this grid here (Tab. 1) as well as other works on the ontology/DSL fusion or reuse. The main comparison criteria are the design domain, the building methodology, the application domain and the technologies and tools. Another comparison grid could be seen in the challenges described by (Walter *et al.*, 2009) regarding 5 challenges, tooling, language interoperability, formal semantics, learning curve and domain analysis (p.1).

#### 3.1 Design Domain and Domain Design

Ontology and DSLs are modelling means to represent a domain. They are used by designers of application in engineering field for the DSL and a more Artificial Intelligence and Knowledge Engineering fields for the Ontology. They are meeting the needs of structuring data and information for application use.

#### 3.2 Building Methodology

According to (Tairas *et al.*, 2009), the ontology building methodology has clearer guidelines than the DSL development methodology. Some useful guidelines for ontology have been published by ((Biebow and Szulman, 1999), (Noy and Mc Guinness, 2001), (Gomez-Perez *et al.*, 2003)). These building methodologies are usually supported by a tool (respectively, TERMINAE, PROTEGE, ONTOWEB). Though, they are based on more informal principles of user needs/requirement gathering as described in RUP (Kruchten, 2001), (Passing, 2006)), or in Object Oriented Design Methodologies in general.

The basic iterative steps to build/develop an ontology/DSL are usually:

- 1) Need and requirement phase: Find what are the users need (from experts, domain documents, definition of the use of the model, explanation of the business processes of the users, and eventually, what's reusable);
- 2) Design phase: It's a knowledge capture and structuring phase where the data collected is organized and structured to be useful within the model;
- 3) Evaluation: the obtained model is evaluated to check whether it satisfies the specification requirements.

For the DSL, the evaluation is based not only on the design result but also on the DSL support tools developed in an 4) Implementation phase, where the necessary tools for executable DSLs are developed (i.e. compiler or an application generator that translates DSL constructs in an existing language for example).

This 4<sup>th</sup> phase is considered as differential in DSL vs. ontologies. The ontology support tools are not considered as part of the ontology design process. However, an ontology is not solely a KB; when it is used, it comes with a set of supporting tools as editors, reasoners, etc.

The building of ontology and developing of DSL both consider as highly useful to reuse existing works. Then in the requirement phase, reusable DSLs or ontologies are envisaged. Several integration methodologies have been developed in the both fields. (Pinto and Martins, 2000), describe several types of integration reusing an existing ontology to build a new one.

- 1) Integration enables to easily collate large ontologies and to reconcile several knowledge sources in keeping their autonomy;
- 2) Merging integration enables to create a unique and consistent ontology;
- 3) Alignment/Mapping integration is done by creating links among ontologies which often have a complementary coverage of a broader domain.

(Mernick *et al*, 2005) identify three patterns of design based on an existing language:

- Piggyback domain-specific features on part of an existing language;
- Specialisation by restricting an existing language;
- Extension: by extending an existing language with new features.

Here are two ways of considering the reuse of existing data but that can be equally used in any type of models.

### **3.3 Application Domain**

As for their origin, ontology and DSL are models used for different application. We mainly recover ontologies in AI application (Classification, Knowledge Base, Dictionaries, and Natural Language Generation) or Web application (Automatic Annotation, Web-Services Orchestration). DSL are found in engineering fields as Systems modelling, Code Generation, Functional and non-functional verification, Simulations...

As their applications are different, their design goals are different. They mainly differ on the automation, security and robustness level they should provide when encapsulated in the final application.

### **3.4 Technologies and Tools**

A lot of applications are affiliated to DSL and ontologies but in two main different ways. We then consider some Specification tools or Meta-tools, used to design/build the (Meta-)models, vs. end-use tools.

The specification tools are on the order of Design tool: Editor, Modeller, Ontology GUI, ontology editor, development framework, model composition tool, development graphical environment, inference engine or Consistency tools: Model checker, reasoning engines... The end-use tools are on the order of an orchestration engine based on an ontology, a Web GUI adapting to the concept of the user, the simulation engine playing a model... They are not specifically based on an ontology or a model and it is on these technologies that we can test the coverage and the semantics abilities of DSL and ontologies modelling the same domain for example.

**Table 1.** Ontology/DSL Comparison grid

Comparison criteria	Ontology	DSL
<b>Design Domain</b>	Knowledge Engineering	Engineering in general (Computer science engineering, System engineering, Electronics - for example: real time embedded systems, robotic systems, avionics systems-)
<b>Building / Development Methodology</b>	Ontology Building <ul style="list-style-type: none"> <li>• User need and requirements capture</li> <li>• Reuse possibilities</li> <li>• Domain knowledge and structuring relations</li> <li>• Evaluation</li> </ul>	DSL development <ul style="list-style-type: none"> <li>• User need and requirements capture</li> <li>• Reuse possibilities</li> <li>• Domain knowledge and structuring relations</li> <li>• Implementation of executable DSL tools (i.e. a compiler or translator, ...)</li> <li>• Evaluation</li> </ul>
<b>Application Domains</b>	Classification, Automatic Annotation, Web-Services Orchestration, Knowledge Base, Dictionaries, Natural Language Generation, ...	Systems modelling, Code Generation, Functional and non-functional verification, Simulations, ...
<b>Technologies and Tools</b>	Ontology GUI, inference engine, ontology editor, reasoning engines, ...	DSL development framework (AMMA, Eclipse GMT,...), Modeler, model composition tool, DSL development graphical environment (DSL Toolkit Microsoft, Papyrus,...), Code generator, Model checker, transformation languages (ATL,...),...
<b>Format</b>	RDF, OWL, Topic Maps,...	MOF, EMF, EMOF, CMOF, SMOF,...

### 3.5 Rationale for the use of ontology in the design process of DSLs

Several works have used ontologies for their semantic or structural complementarity with DSLs (with the limitations underlined in the previous section). Two main types of ontologies are referred; (1) the ontology as the *explicit specification of a conceptualization* (Morin *et al.*, 2009), (Walter, 2009) and (2) the Ontology as the metaphysical study of the nature of being and existence (Kurtev, 2007). (Kurtev, 2007) is designing a meta-language (OGML) based on the

metaphysical principles of Ontology. It results in a high-level DSL close to the formal ontology metamodels. (Morin *et al.*, 2009) and (Walter, 2009) are combining the ontologies with DSL at a meta-level to extend the coverage of a DSL (Walter and Ebert, 2009) or to integrate a variability viewpoint straightforwardly in a DSML (Morin *et al.*, 2009). In the PROTEUS project, the ontology as (2) is firstly used to represent the domain, i.e. inferring information from a KB that complements with the Architecture DSL and then to develop the DSLs, i.e. as a representation of experts' knowledge.

## **4 On the use of ontology for the development of the PROTEUS DSLs**

### **4.1 The Proteus Project**

In (PROTEUS, 2009), the robotic market is described as existing and going on growing at a fast pace. It is then of utmost importance to help French industries have their share of it. The PROTEUS project (Plateforme pour la Robotique Organisant les Transferts Entre Utilisateurs et Scientifiques meaning Robotic Platform to facilitate transfer between Industries and academics) goal is to create a portal for the French robotic community. Such a portal, to be of use, will be constituted of many parts and one key point is a toolset, the technological part of the platform.

The PROTEUS platform will enable to gain new skills in order to create or improve new products, new process or new services by providing easier way to collaborate inside a community. The software created through the project will include components and architecture description allowing its users to create complex systems where they should be able to assess and validate generic software technologies. Moreover, the link to real robots will allow these generic technologies to be inserted in hardware.

The shared infrastructure provided will take into account the definition of the so called standardized robot architecture for some specific domain as well as generic capabilities through the use of formal representations associated with generator tools. The platform should enable the community to use real robots operated by end-users in order to directly assess its achievements (e.g. cognition and control algorithms) onto real industrial robots. The software-oriented considerations are taken into account through tools facilitating knowledge transfer, executable environments creation, and methodologies to make these enabling resources easily exploitable by diverse and numerous adopters of the community. The work to be done on this axis will be to provide a minimal formal language to support the description of scenarios and model integration facilities (model means here an external component, either stand alone components or library of components that that provides access point and capability to be externally sequenced) and open simulation architecture.

The robotic fields considered are restricted to mainly aerial and terrestrial robotic as well as considerations on humanoid robotic.

## 4.2 The PROTEUS robotic Ontology

The robotics ontology within PROTEUS is seen as a tool for modelling and analyzing robotic systems. Development, test and validation of embedded systems like mobile robots involve different knowledge domains. A robot's physical structure and control software are designed in order to fulfil a given type of missions. Testing the robot's functionalities needs a model of a robot as well as a model of its nominal environment. Ontologies can be used to do this modelling and validate it (PROTEUS, 2009).

The ontology has been built from the knowledge of robotics experts involved in different sub-domain of the field. Their expertise concerns the following domains: command, control, perception, navigation, localization, traffic control, optimization, mission planning to simulation. To share their knowledge and models it, the experts have been brought together in *modelling meetings*. Their exchanges were supported by the description of scenario representing the use of their field within the autonomous systems domain.

From these modelling meetings the requirements of the Robotic ontology in PROTEUS were described. The Robotic ontology should be able to model the mechanical and electronic component models, as well as control architectures, consistency detection systems and simulators, database of components, control tasks or complete subsystems. The ontology should help modelling different versions of a given robot and follows its evolution. As it comes to model the robot's behaviour, its ability to perform the missions for which it was designed; there is also a need for modelling its environment and its mission in urban area (PROTEUS, 2009).

The resulting OWL DL ontology (in its primary version) consists of 205 concepts linked with 73 relations (OWL properties). A screenshot of this ontology is presented in Fig.1. The ontology is organized around a kernel ontology describing the main concepts of the field and extended with several sub-domain ontologies describing the following topics: Robotic Components, Information, Mission, Environment, and Simulation.

This first version of the robotic ontology has been validated by the industrial users involved in the PROTEUS project but extensions' refinement is planned for the follow-up of the PROTEUS project.

In the PROTEUS project, the ontology is used not only to represent the domain and to develop the Robotic Architecture DSL as presented in this paper, but also to validate DSLs manually developed on Communication and Algorithm.

And then it is also plan to use this ontology:

- 1) To normalize the robotic domain;
- 2) To support inference at run-time;
- 3) To automatically transform Ontology in DSL.



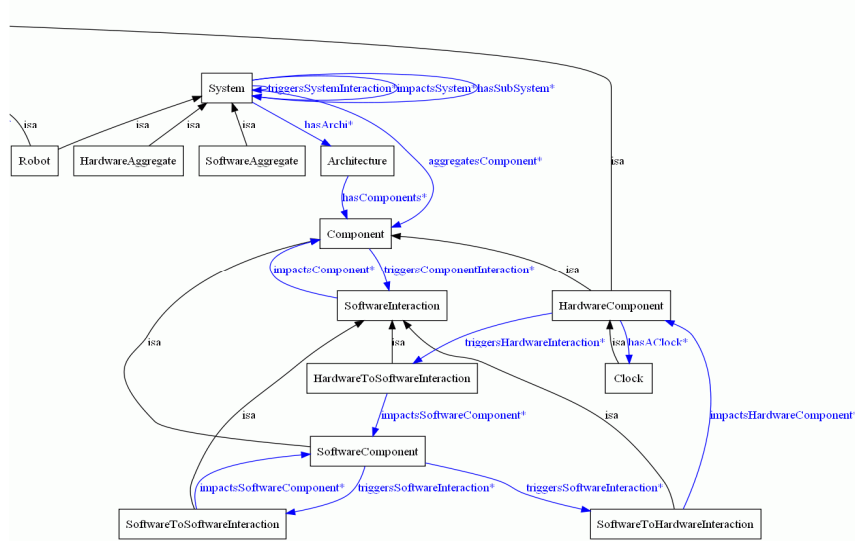


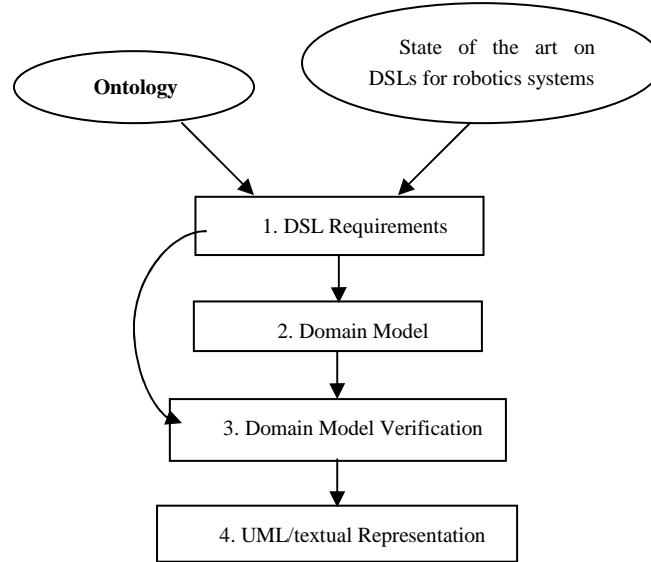
Fig. 1. Screenshot of a part of the kernel ontology

### 4.3 From PROTEUS ontology to PROTEUS DSLs

#### 4.3.1 Rationale of the ontology use in the Architecture DSL design process

The methodology we followed in PROTEUS is described in Fig.2. We show the DSL design process that integrates the ontology. The design process consists in four steps:

1. The requirements of the DSL are gathered from both following sources: in the one hand from the ontology and in the other hand from the state-of-the-art on DSL for robotics systems.
2. Building the domain model of the DSL: The purpose of the domain model is to describe formally the concepts of the domain. The domain model will be described by the means of one or more class diagrams, as well as in the form of textual descriptions.
3. Domain model verification: this step is intended to verify that the aforementioned domain model is covering all the requirements expressed in the first step.
4. UML/textual representation: An alternative for the specification of a DSL is the use of UML, which is a widely known modelling language that has a lot of support tools (Giachetti *et al.*, 2009). Another alternative is a textual representation of the DSL.



**Fig. 2.** Integrating the ontology in the design process of the DSL

The ontology is involved in the steps 1 and 2 of the DSL design process. Indeed, the first requirement of the DSL is to correspond to domain concepts defined in the ontology. The other requirements, coming from the ontology, are derived from this one. From the ontology, we extract all the concepts that are specific to the domain. Those concepts are then filtered to retain only the relevant ones for the DSL. On the other hand, if some concepts are missing in the ontology, they are added to the domain model of the DSL.

**Table 2.** Mapping the ontology to the DSL domain model

Ontology (OWL)	Domain model (UML class diagram)
Concept	Class
subClassOf	Generalization
Property	Association
Property:IsA	Inheritance
Property:HasA	Composition
Cardinality	Multiplicity

The table 2 shows the transition from the ontology, written in OWL DL language, to the DSL domain model specified as a UML class diagram. OMG also proposes the ODM (Ontology Definition Metamodel) which defines a set of QVT mappings from UML to OWL (IBM *et al.*, 2009). Only the automatic transformation from UML to OWL is implemented, the transformation from OWL to UML is not implemented yet.

So we have not taken advantage of the ODM project to make the transition from OWL to UML.

#### 4.3.2 Architecture DSL development on Robotic ontology based requirements

One of the objectives of the PROTEUS project is to provide domain specific languages (and related tools like editors, consistency checkers, etc ...) suitable to specify missions, environments and robot behaviours that have been specified by robotics experts involved in the project. The discussions under the PROTEUS project have lead to the decision of defining three DSLs:

1. The "Architecture DSL" which will ease the definition of specific robotic architectures (reactive, deliberative, hybrid) and specific components that form those architectures (sensors, actuators, planners).
2. The "Control & Communication DSL" that will control the robotic components and will ease the definition of communication mechanisms between components (sending/receiving of events and data).
3. The "Algorithms DSL" that will ease the definition of algorithms which are to be used, triggered with the "Control & Communication DSL" for implementing behaviours in the different components of an architecture described with the "Architecture DSL".

To assist the development of the PROTEUS DSLs, we have used the ontology presented in section 3.2 to build the domain model of the DSLs. In this section, we only present the case of the "Architecture DSL".

The main entry of the design process of PROTEUS DSLs is the ontology presented in section 3.2. As stated before, the ontology is organized around a kernel ontology and extended with several sub-domain ontologies describing the following topics: Robotic Components, Information, Mission, Environment and Simulation.

From this ontology, relevant concepts related to robotic architecture are extracted. In table 3, we list some of those relevant concepts:

**Table 3.** Subset of concepts extracted from the PROTEUS ontology

Concept	Ontology Source	Concept	Ontology Source
Component	kernel	WeaponHardware	components
SoftwareComponent	Kernel	PowerHardware	components
HardwareComponent	Kernel	SensorHardware	components
Environment	kernel	EnvironmentParameterSensor	components
ActuatorHardware	components	ImageSensor	components
PhysicDevice	components	LocalizationSensor	components
MotorizationHardware	components	ObjectDetectionSensor	components
PrehensionHardware	components	ObjectTrackingSensor	components

From the set of concepts that we have associated to the domain model of the "Architecture DSL", we have eliminated some concepts that are not relevant for the definition of the DSL. For example, the concept "Architecture" is too general and is useless for the specification of a robotic architecture.

On the other hand, we have noticed that there are some missing concepts that have to be added to the domain model of the DSL. For example, concepts such as "ComputingHardware" and "StorageHardware" are not contained in the first version of this ontology and are compulsory for an Architecture DSL.

## 5 Conclusion

In this article, we have presented a methodology for reusing ontologies in the development process of domain specific languages. The methodology is used in the case of the PROTEUS project and has lead to the definition of the specification requirements for the Robotic Architecture DSL of PROTEUS. The algorithm DSL and the control/communication DSL will be developed from scratch but a comparison methodology will be set up for validation on the ontology.

As an ontology is used more to classify things of the World and DSL are more used to build engineering artefacts, another work to be done is to handle an ontology guided approach to delimitate the borders of these three DSLs from the ontology. Is there tracks in the ontology model of the design viewpoint of the architect? Is it possible to delimitate the scope of the architecture viewpoint? Isn't a DSL an engineering viewpoint on a larger domain in itself?

## 6 Acknowledgements

We want to thanks Sébastien Madenat, Kévin Le Noir and Florian Noyrit for their fruitful discussions and enlightened comments (we remain responsible for what is written in this paper).

## References

- ATL, (2010), <http://www.eclipse.org/atl/>
- Bézivin J., (2004), Sur les principes de base de l'ingénierie des modèles. RTSI-L'Objet, 10(4) : Page 145-157, 2004. <http://atlanmod.emn.fr/www/papers/LObjet2004.pdf>
- Biébow B, Szulman S., (1999), TERMINAE: a linguistic-based tool for the building of a domain ontology. In *EKAW'99 Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and management*. Dagstuhl, Germany, LCNS, pages 49-66, Berlin, 1999. Springer-Verlag
- Biezunski, M., Bryan, M., et Newcomb, S. R., (1999), *Topic Maps, specification ISO/IEC 13250*, 3 December 1999.

- Blanc, X., Mougenot, A., Mounier, I., and Mens, T., (2008), Detecting model inconsistency through operation-based model construction, in Robby, editor, *Proc. Int'l Conf. Software engineering (ICSE'08)*, volume 1, pages 511-520. ACM, 2008.
- Brickley, D., et Guha, R.V., (2004), *RDF Vocabulary Description Language 1.0: RDF Schema* <http://www.w3.org/TR/rdf-schema/>.
- Buitelaar, P., Cimiano, P., Magnini, B., (Eds.), (2005), *Ontology Learning from Text: Methods, Evaluation and Applications*, in *Frontiers in Artificial Intelligence and Applications Series*, Vol. 123, IOS Press, July 2005.
- Caplat, G., (2008), *Modèles et métamodèles*, presses polytechniques et universitaires romandes, INSA Lyon, 197p.
- Cimiano, Ph., and Völker, J., (2005), Text2Onto, A framework for Ontology Learning and Data-Driven Change Discovery, in *Natural Language Processing and Information Systems*, LNCS, 2005, pp. 227-238.
- Falbo, R.A., Menezes, C.S., and Rocha, A.R.C., (1998), A Systematic Approach for Building Ontologies. In *Proceedings of the 6th Ibero-American Conference on Artificial Intelligence*, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.
- Gašević, D.O., Djurić D.O., Devedžić V.B., (2005), Bridging MDA and OWI Ontologies, *Journal of Web Engineering*, Vol. 4, No.2, pp. 118-143,
- Giachetti, G., Marín, B. and Pastor, O., (2009), Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles, in *Lecture Notes in Computer Science*, 2009, Volume 5565/2009, 110-124, DOI: 10.1007/978-3-642-02144-2\_13
- Gómez-Pérez, A., Manzano-Macho, D., Alfonseca, E., Núñez, R., Blacoe, I., Staab, S., Corcho, O., Ding, Y., Paralic, J., et Troncy, R., (2003), Ontoweb, *Deliverable 1.5: A survey of ontology learning methods and techniques*, IST Project IST-2000-29243 OntoWeb, June 2003.
- Gruber, T., (1993), A translation approach to portable ontologies, in *Knowledge Acquisition*, 5(2), pp. 199-220.
- Gruber, T., et Lytras, M.D., (2004), *Interview for the AIS Special Interest Group on Semantic Web and Information Systems*, Volume 1, Issue 3, 2004.
- Gruber, T.R., (1995), Toward principles for the design of ontologies used for knowledge sharing, in *International journal of human-computer studies*, vol. 43, no 5-6, pp. 623-965, 1995.
- Guizzardi, G., (2007), On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*, Olegas Vasilecas, Johan Edler, Albertas Caplinskas (Editors), ISBN 978-1-58603-640-8, IOS Press, Amsterdam, 2007.
- Handschuh, S. & Staab, S., (2003), Annotation for the Semantic Web, IOS Press, in *Frontiers in Artificial Intelligence and Applications*, Vol. 96, 2003, 240p.
- International Business Machines, Object Management Group, and Sandpiper Software. Ontology definition metamodel (ODM). OMG Document ad/2003-02-23. Available at <http://www.omg.org/>, 2009.
- Kruchten, Ph., (2003), the Rational Unified Process, Object Technology Series, Booch, Jacobson, Rumbaugh (Eds), Addison-Wesley, 332 pages.
- Kurtev, I., (2007), Metamodels: Definitions of Structures or Ontological Commitments? *Workshop on TOWERS of models*. Collocated with TOOLS Europe 2007, p. 53-65 <http://www.cs.york.ac.uk/ftpd/ir/reports/2007/YCS/416/YCS-2007-416.pdf>
- Lassila, O., et McGuinness, D., (2001), The Role of Frame-Based Representation on the Semantic Web, in *Linköping Electronic Articles in Computer and Information Science*,

- Issue: Vol. 6 (2001) : 005, <http://www.ida.liu.se/ext/epa/ej/etai/2001/018/01018-etaibody.pdf>
- Lemesle R., (1998), Transformation rules based on meta-modeling, EDOC'98, San Diego, November 3-5, 1998, <http://atlanmod.emn.fr/www/papers/EDOC98-lemesle.pdf>
- McGuinness, D.L., et Van Harmelen, F., (2004), *OWL Web Ontology Language Overview, W3C Recommendation* 10 February 2004, <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- Mernik, M., Heering, J., and Sloane, A.M., (2005), When and how to develop domain-specific languages, in *ACM Computing Surveys*, 37(4), pp. 316-344, December 2005.
- Morin B., Perrouin G., Lahire Ph., Barais O., Vanwormhoudt G., and Jézéquel J-M., (2009), Weaving Variability into Domain Metamodels, in *ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09)*, Denver, Colorado, USA, Oct 2009. <http://www.irisa.fr/triskell/publis/2009/Morin09c.pdf>
- Noy, N., and McGuinness, D., (2001), *Ontology Development 101: A Guide to Creating Your 1st Ontology*, [http://protege.stanford.edu/publications/ontology\\_development/ontology101-noy-mcguinness.html](http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html)
- Noy, N., Musen, M.A., (2000), PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, *AAAI/IAAI 2000*: 450-455
- OMG, Meta Object Facility (MOF) Core Specification Version 2.0, 2006, <http://www.omg.org/mof/>
- Passing, J., (2006), Requirements Engineering in the Rational Unified Process, Seminar Requirements Engineering, Summer term 2006, Hasso Plattner Institute for Software Systems Engineering, 16 pages.
- Pinto, H.S., and Martins, J.P., (2000), Reusing Ontologies, In *AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, AAAI Press, pp.77-84 <http://www.aaai.org/Papers/Symposia/Spring/2000/SS-00-03/SS00-03-012.pdf>
- PROTEUS Consortium, (2009), *PROTEUS- - Platform proposal to ANR 2009 ARPEGE call*, 152 pages.
- QVT, (2008), (OMG-QVT 2.0, 2008) <http://www.omg.org/spec/QVT/index.htm>
- Tairas, R. and Mernik, M. and Gray, J., (2009), Using Ontologies in the Domain Analysis of Domain-Specific Languages, in *Models in Software Engineering*, Lecture Notes in Computer Science vol. 5421, Chaudron, M. (Ed), Springer Berlin / Heidelberg, pp.332-342.
- Uschold, M., et Grüninger, M., (1996), Ontologies: Principles, Methods and 16 applications, in *Knowledge engineering review*, vol.11, N.2., p. 93-136.
- Walter, T. and Ebert, J., (2009), Combining DSLs and Ontologies Using Metamodel Integration, in *Domain-Specific Languages, Lecture Notes in Computer Science n°5658*, Taha, W. (Ed), Springer Berlin / Heidelberg, p. 148-169
- Walter, T., Silva Parreiras, F., and Staab S., (2009), OntoDSL: An Ontology-Based Framework for Domain-Specific Languages, in Schürr, A., and Selic, B., (Eds) *Lecture Notes in Computer Science*, vol. 5795, Springer Berlin / Heidelberg, pp.408-422
- Weisemöller, I., Schürr, A., (2007), A Comparison of Standard Compliant Ways to Define Domain Specific Languages, in *MoDELS Workshops*, Lecture Notes in Computer Science, Vol. 5002, pp. 47-58, Springer, 2007.

# Model-based design of Intelligent Mobile Robot

Takahiro TAKASU\*, Tsunehiko FUJITA\*, Yusuke ZAMA\*, Koji ISHIDA\*,  
Makoto MIZUKAWA\*, Yoshinobu ANDO\*, Takashi YOSHIMI\*,  
Takeshi SAKAMOTO\*\*

\*Shibaura Institute of Technology, 3-7-5, Toyosu, Koto-ku, Tokyo

\*\*Technologic Arts Incorporated

[m109056@shibaura-it.ac.jp](mailto:m109056@shibaura-it.ac.jp)

**Abstract.** We are designing and developing an autonomous mobile robot based on model-based engineering approach. In this paper, we will discuss features of reusability of existing robot modules through this development.

**Keywords:** Mobile Robot, RT-Middleware, SysML

## 1 Introduction

In Japan there are a growing number of disabled and elderly people who usually require personal transportation vehicles, such as car, bicycle and wheelchair. However, currently people have many difficulties or are unable to use these vehicles in complex environments such as shopping mall or amusement center.

The NEDO's "Intelligent RT Software Project"<sup>[1]</sup> was started to promote the Robot Technology (RT) as the basic technology to solve various problems in real life. One of the purposes of this NEDO project is the development of intelligent mobile robot for providing personal mobility. In the scope of this project, we aim to develop the versatile intelligent RT modules which are able to provide safe and convenient transportation service for disabled people to create barrier-free society. We also try to make the development of intelligent mobile robot easier by applying RT technology.

The National Institute of Advanced Industrial Science and Technology (AIST) developed RT-Middleware to solve efficiency of the development of robot [2]. RT-Middleware modularized robot functional component and defined its structure as RT-Component. The project has been actively implementing RT-Component for efficient robot development.

Some personal mobile robots that adopted RT-Components have been developed under the supervision of NEDO project<sup>[3][4]</sup>. However, these robots have not fully realized the model-based design process. Therefore the resources from these researches have some difficulties to reuse.

If we achieved to an intelligent mobile robot with model-based design, the robot software components can be reused in other researches easily.

## 2 Purpose

This paper presents the design and development of our model-based intelligent mobile robot. We adopt model-based design to make robot models independent from specific hardware and software platform so that other developers will be able to refer our models and customize it to meet their robot specification.

We make use of existing RT-Components which have been developed by the NEDO's "Intelligent RT Software Project" in this research to reduce our development effort, as well as to extend the reusability of RT modules. We also employ the extended and common interfaces of RT-Component in mobile robots which were proposed in the working group of the "Intelligent RT Software Project" [5]. These interfaces cover standard specification of robot position information defined in OMG Robotic Localization Service (RLS) [6]. By using these standard RT-Components and interfaces, our proposed robot model will be more versatile.

## 3 Mobile Robot

We are developing autonomous mobile robot for transporting passenger to some designated positions. Our mobile robot will have to provide following functions as the minimum requirement for the mission.

1. Autonomous navigation to target position.
2. Ensuring safety of passenger and surrounding environment and people.
3. Allowing control by passenger

Additional requirements are discussing on requirement diagrams.

## 4 Models

This paper shows the progress of our model-based design. We adopt the international standard modeling language SysML (OMG System Modeling Language)<sup>[7]</sup> in our design process. The following diagrams in this paper are made of SysML.

### 4.1 Context diagram

Our mobile robot will operate in outdoor environment with pedestrians, bicycles, cars, and in different road conditions such as grass, steps, slopes, and so on. The robot must be able to avoid the obstacles and make correct path planning in different situations. Figure 1 shows our expecting objects, environment, and disturbance elements. Disturbance elements are the possible noise sources that may affect the robot's sensing ability.



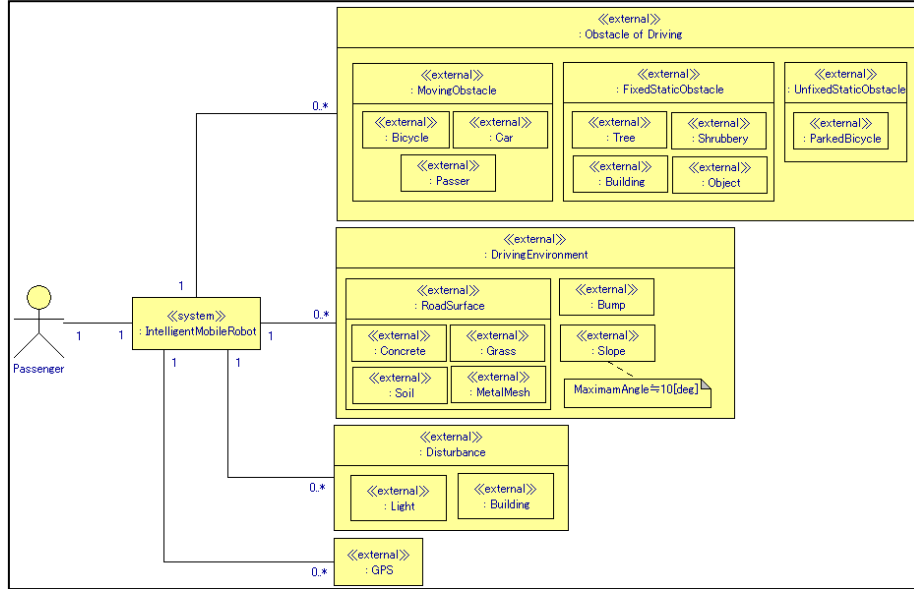


Fig. 1 Context diagram

## 4.2 Requirement diagrams

Our system requirement is described in the three following diagrams. Our models focus on the requirements of mobility, safety of operation and extending of RT modules reusability.

Figure 2 shows the basic requirements, namely “Safety Locomotion” and “Extending RT-Module Reusability”. These are the top-level requirements. “Safety Locomotion” is composed of the requirements about locomotion, which are “Controlled by Passenger”, “Autonomous Locomotion” and “Enhanced Safety”.

Figure 3 shows the requirements and functions of robot locomotion. Most of them are requirements for self locomotion. These requirements are represented from abstract levels to the concrete description with device or software block.

Figure 4 shows the requirement of the safety of operation. We discussed the requirement for robot’s function to avoid dangerous situations, such as collision, malfunction and sudden break avoidance, or passenger and environment’s safety. Our mobile robot meets these requirements by the corresponding software modules and “Emergency Lamp” notification.

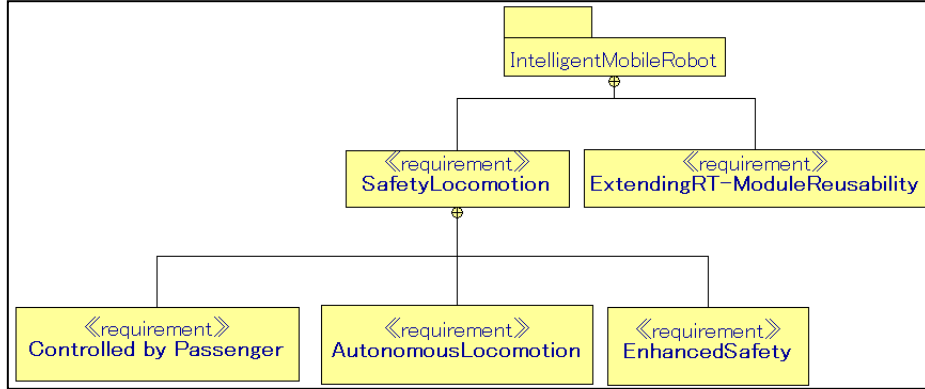


Fig. 2 Requirement diagram

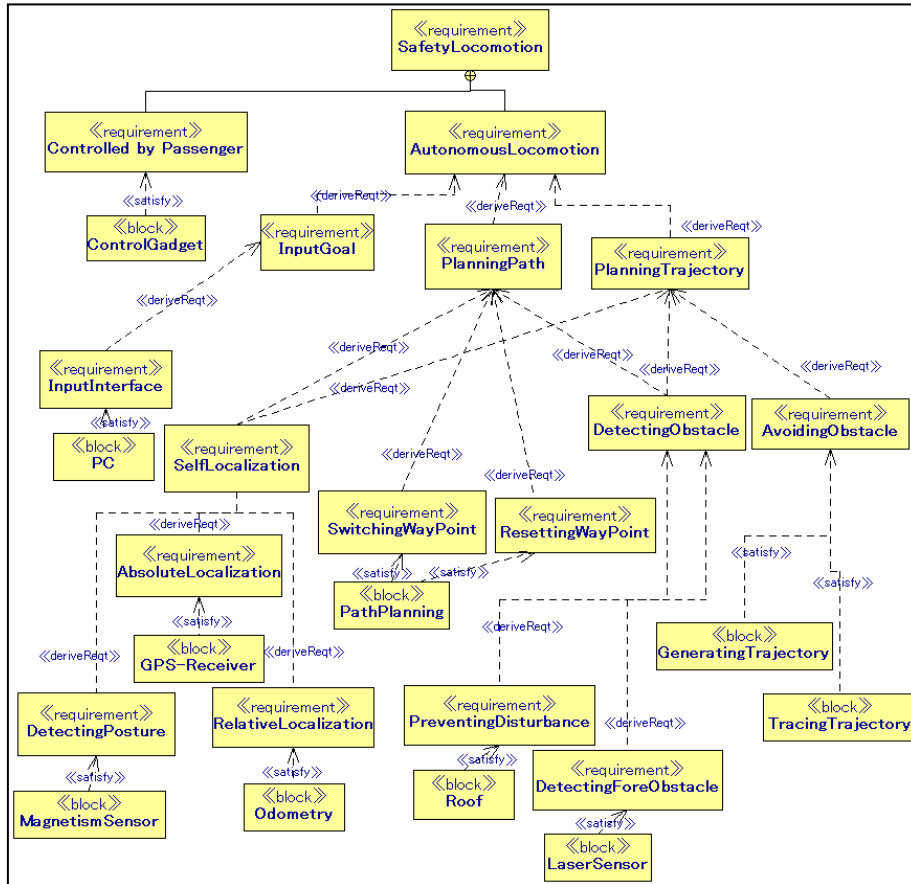


Fig. 3 Requirements of autonomous locomotion

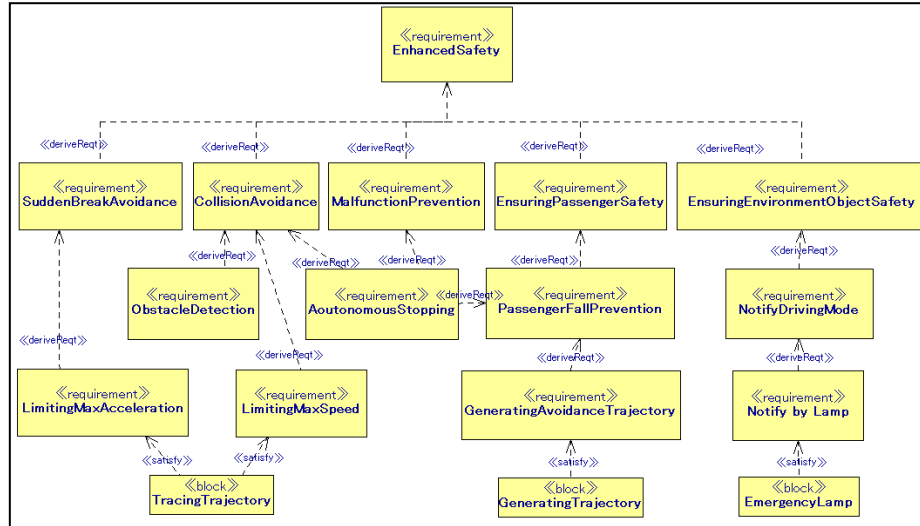


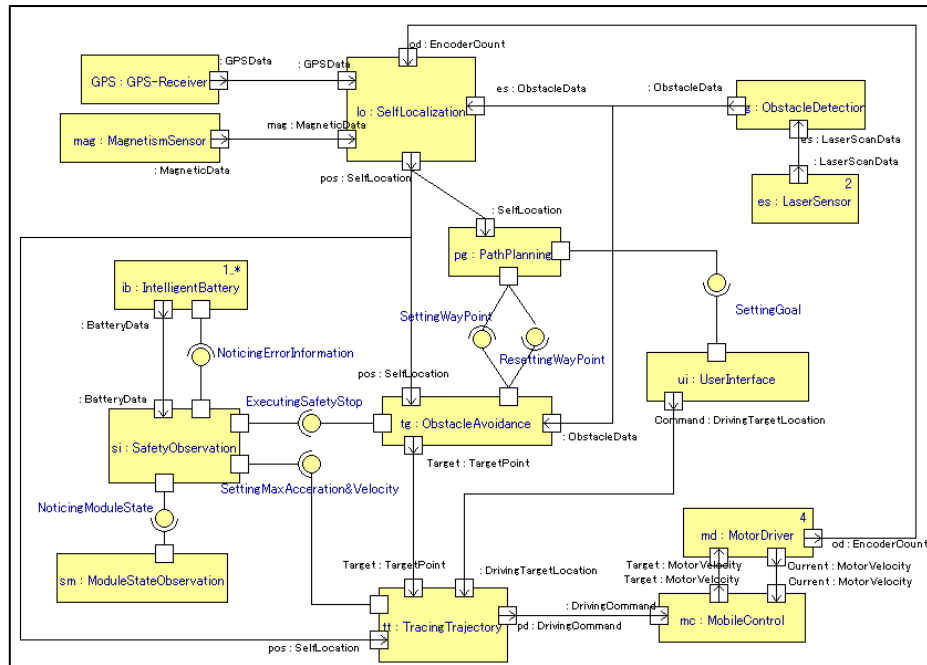
Fig. 4 Requirements of safety

### 4.3 Structure

We separated the required components of our mobile robot, and decided the interfaces between these components. By implementing the common interfaces that were proposed by the working group of the “Intelligent RT Software Project”, our mobile robot can easily adopt and apply existing RT-Components.

In addition, our models separated device specific components from components for transporting functions. Thus our models are independent from specific device (especially wheeled platform), and are able to be reused easily.

Figure 5 shows Internal Block Diagram (IBD) of our robot’s software structure. Figure 6 shows Block Definition Diagram (BDD) of our robot’s hardware structure.



**Fig. 5 Diagram of software structure**

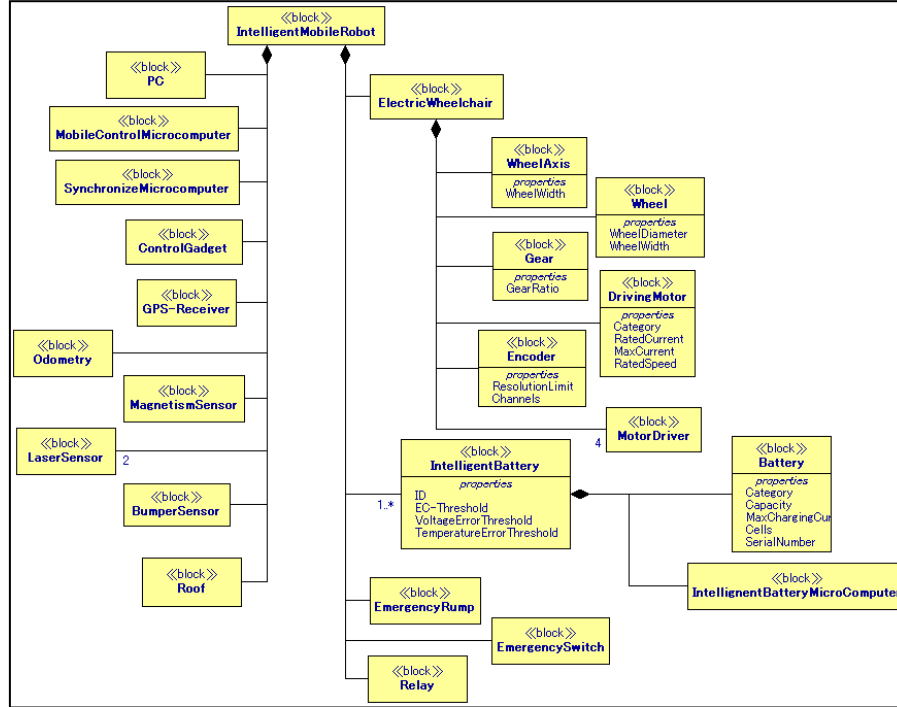


Fig. 6 Diagram of hardware structure

## 5 Conclusion

In this paper, we have presented the basic specification of our mobile robot. The proposed mobile robot design is fully based on model-based methodology to overcome the current difficulties of mobile robot development. Highlighted features of our approach are straightforward, platform independent and reusable model design, reduced development effort and customizable system configuration.

Based on this basic specification, future work will focus on the detailed design for each module. Then we will select the suitable RT-Components from the RTC Center of “Intelligent RT Software Project”.

Besides, we will further extend the reusability of the platform-dependent component, such as the “Mobile Control” by extracting the platform-independent parts.

## 6 Acknowledgement

This work is supported by NEDO (New Energy and Industrial Technology Development Organization, Japan) project “Intelligent RT Software Project”.

## 7 References

- [1] Katsuya OKANO, Yusuke YASUKAWA, NEDO: “Overview of Intelligent RT Software Project”, (The 27<sup>th</sup> Annual Conference of the Robotics Society of Japan, RSJ2009AC1D1-01, 2009)
- [2] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. Yoon: ”RT-middleware: distributed component middleware for RT (robot technology)”, (In IEEE/RSJ Int. Conf. on robots and intelligent systems, pp. 3555–3560, 2005)
- [3] Yusuke HIEIDA, Kentaro TAKEMURA, Akihiro ITO, Junichiro KUWAHARA, Tsuyosi SUENAGA, Jun TAKAMATSU, Tsukasa OGASAWARA, Nara Institute of Science and Technology (NAIST): “Study Design Guidelines of Components for Mobile Robot Using RT-Middleware”, (Robomec2010, 2P1-A20, 2010)
- [4] Eijiro Takeuchi (Tohoku Univ.), Yamazaki Masafumi (Tohoku Univ.), Tanaka Kazushi (Tohoku Univ.), Kazunori Ohno (Tohoku Univ.), Satoshi Tadokoro (Tohoku Univ.), Saitoh Toshihisa (Segway Japan, Ltd.), Hiroki Igarashi (Kyoto Univ.), Fumitoshi Matsuno (Kyoto Univ.), and Toshi Takamori (International Rescue System Inst.):  
 “Development of RTCs for Mobile Robots with Autonomy and Operability  
 Report 14: Distributed Processing for Autonomous Navigation System using RT-Middleware”,  
 (SICE System Integration 2009, SI2009, 3B2-1, 2009)
- [5] Masaharu SHIMIZU (CIT, fuRo), Nobuyuki KITA (AIST), Toshihisa SAITO (Segway Japan), Eijiro TAKEUCHI (Touhoku Univ.), Yusuke NAKAJIMA (AIST), Naohito TAKEGAWA (AIST), Hiroki IGARASHI (Kyoto Univ.) Yasuo HAYASHIBARA (CIT), Hideaki YAMATO (CIT, fuRo), Kengo TODA (CIT, fuRo), Takayuki FURUTA (CIT, fuRo), Makoto MIZUKAWA (SIT):  
 “The Joint Interface of RT Component for Mobile Robots  
 - The Second Activity Report of Mobile Robot One Sub WG in NEDO Intelligent RT Software Project -”, (SICE System Integration 2009, SI2009, 3D2-1, 2009)
- [6] Object Management Group: “OMG Robotic Localization Service”,  
[\(http://www.omg.org/spec/RLS/\)](http://www.omg.org/spec/RLS/)
- [7] Object Management Group: “OMG System Modeling Language”,  
[\(http://www.omgsysml.org/\)](http://www.omgsysml.org/)

# Model Driven Embedded Systems Design Environment for the Service Robotics Domain

Martijn Rooker

PROFACTOR GmbH  
Im Stadtgut A2  
A-4407 Steyr-Gleink, Austria  
`martijn.rooker@profactor.at`

**Abstract.** The paper describes a multi-domain model-driven embedded systems design approach. The main target for this approach is the domain of complex Industrial Automation and Control Systems (IACS). The special requirements of the industrial automation sector are taken into account by the novel approach, utilizing existing model-driven techniques. One of the specific target areas is the field of Service Robotics. This paper will describe the approach and introduce a use-case where robotics modules will be configured for customer tailored applications in the service robotics domain. This approach is currently being develop in the Framework Seven (FP7) Embedded System Design project MEDEIA funded by the European Commission.

**Keywords:** Model-driven development, Automation Component, Service Robotics

## 1 Introduction

Today's production processes will be performed more and more by automated machines and robots as the level of automation increases steadily. The main trend in industrial automation is the increasing need for customized and individualized plants. The production lines have to be constructed and adapted to new production processes quickly [1]. Such highly automated systems are mainly controlled by embedded hardware and software which are heterogeneous in nature, which lead to a tremendous increase in the design complexity. The software engineering costs will increase up to 80% of the overall system costs in the next 15 years (currently this ratio is about 55%) [2][3].

One of the main current design trends in automation and control systems is to put components, blocks made of hardware, software and intellectual property together (like algorithms and data structures)[4]. This in turn calls for common, language independent models for representing, saving and reusing such components. Neither the state-of-the-art or current trends in design and engineering in industrial automation [1][4] are capable of providing an applicable solution to the problem.

The current situation for the design of automation and control systems is characterized by a huge variety of different approaches in both the specification and

implementation of plants and systems. Different end-users of an automation system may use any of the available specification methods, often dependent on their domain. The specification method is used to describe end users' needs and wishes. End user specifications can take several forms, e.g. textual descriptions, Pipe and instrumentation diagrams, list of sensors and actuators, etc.

When different end-users have to interact, difficulties arise. Every end-user will use its own description method of specifying its needs for the system. Currently, very little support exists in automated workflows or even in the interoperability of different software tools implementing the combined set of specifications, partly because the specifications themselves have been defined in different forms. The problems are not only limited to the front end of the development process. In any automation solution, the implementation of solutions reflecting end-user requirements is completed by programmers each of whom expresses the solution based upon different preferences according to the type of plant and various environment conditions.

The paper is organized as follows: section 2 provides an overview of the MEDEIA approach. The MEDEIA Automation Component Meta-Model is explained in more detail in section 3. Section 4 will provide a use-case where the MEDEIA approach is applied to the area of service robotics. Finally, Section 5 will conclude the paper and talks about future work with MEDEIA.

## **2 Model-Driven Design for Industrial Automation and Control Systems**

One of the main obstacles for efficient engineering of control systems is that the various design methodologies as well as the implementation technologies within plants are diverse according to different available solutions. This aspect makes the overall design of automation solutions very difficult, especially when different machines, robots and embedded devices have to work together. To overcome these limitations of high diversity, the MEDEIA approach aims at a meta-design architecture for the plant architecture with the following objectives:

- *Formal framework for model-driven component-based development of embedded control:* The main focus in the MEDEIA project lies on a formal framework for the design of heterogeneous embedded automation and control systems. The main components of the framework are Automation Components (AC), which in general are a combination of embedded hard- and software.
- *Easily understandable modeling method applicable for domain experts:* The focus is to use modeling methods from different domains (e.g. software engineering, control engineering in manufacturing, etc.) to define a solution that can be applied in the target areas.
- *Integrated modeling of diagnostics:* An integrated diagnostics concept will be developed that analyses faulty and out-of-norm behavior to effectively support the maintenance process of heterogeneous embedded hard- and software.



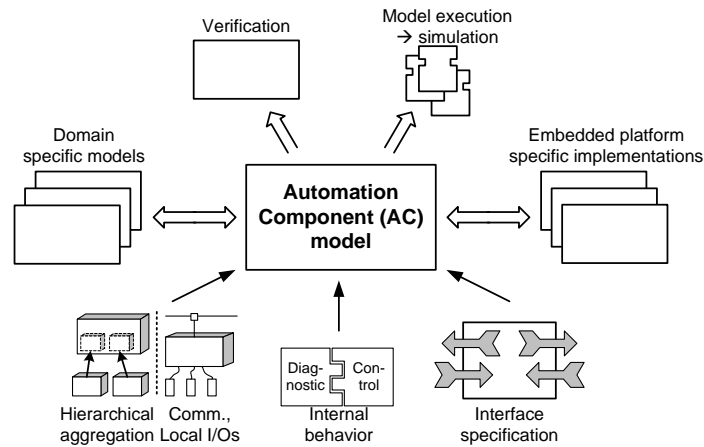
- *Integrated simulation and verification of systems design*: An integrated simulation and verification concept will be developed that enables different configurations of the system architecture, including hardware-in-the-loop to be simulated.
- *Automatic, embedded platform specific code-generation*: Automation Components can be deployed to heterogeneous automation hardware through an automatic code-generation, based on the Automation Component Model and the System Model.

## 2.1 General Approach

As mentioned above, the MEDEIA design methodology is based on the so-called Automation Components (AC) which are, generally speaking, a combination of embedded hard- & software. The start of the engineering flow is the definition/specification of the system requirements.

## 2.2 Automation Component Model

The determining element within the MEDEIA approach is the Automation Component (AC) – a combination of embedded hard- and software. The overall plant or automation application is represented as an AC as well as each single device on the lowest level. The plant evolves by the aggregation of ACs to more complex ones, or it is split up into smaller ACs. The abstract meta-model provides the core of each AC. Fig. 1 depicts the architecture of an AC.



**Fig. 1.** Automation Component (AC) Model

The MEDEIA AC Model is represented by use of the following characteristic elements:

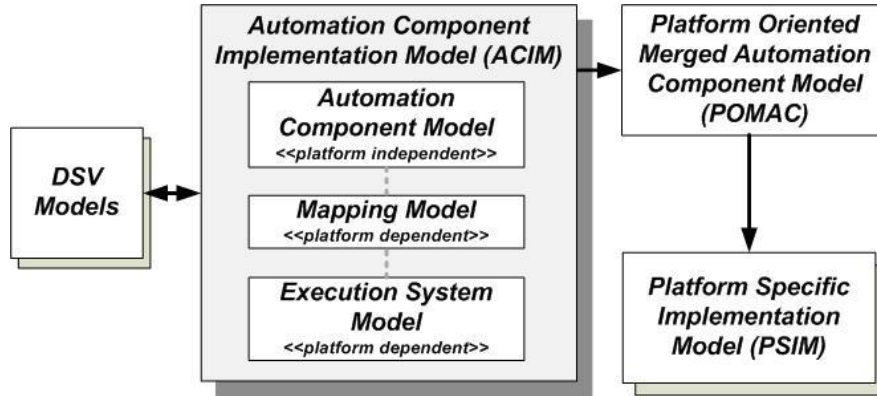
- *Interface description:* An AC is represented by its interface to the environment. MEDEIA enforces a detailed interface description, which includes not only data and action ports, but also timing behavior.
- *Internal behavior:* In addition to the interface description, MEDEIA also addresses the description of the internal behavior of the ACs, summarizing into a stateful description of the AC.  
*Hierarchical aggregation:* As the MEDEIA approach is based on a hierarchical structure of a plant, each AC will use ACs one level below.
- *Communication / local I/Os:* As soon as an AC is related to an embedded hardware device, additional information is obtained from the communication and the associated local I/Os.
- *Domain specific models:* Supporting different domain specific models, the system provides the design engineer the possibility to understand, design and engineer ACs in his familiar way.
- *Embedded platform specific implementations:* By the use of model transformations, an AC can be transferred to be executed on real hardware and also predefined components can be integrated into the general AC model.
- *Model execution ↔ simulation:* MEDEIA provides a transformation of the AC model into an executable simulation framework, providing a flexible basis for the integration into existing plant areas.
- *Verification:* In addition to simulation, the verification of components by themselves and their aggregation provides a second basis for increased engineering efficiency. Therefore, the transformation of the AC model into a formal description of a well known verification approach is included too.

### 3 The MEDEIA Automation Component Meta-Model

There are three elements within the MEDEIA architecture that describe the main idea behind the engineering approach, namely the Domain Specific View (DSV), the Platform Specific View (PSV) and the Automation Component Model (ACM). The DSV is the view from the end user and is determined largely by special requirements and specifications, often unique to the domain itself. The PSV is a developers perspective that varies insofar as there are often several different means to achieve a given functionality. The ACM is the common element that links these two elements. Any relevant information specified within a DSV will be transformed into the ACM and based on the information available in the ACM, at least the rough architecture of source code for a given platform will be generated automatically.

#### 3.1 Automation Component Implementation Model (ACIM)

A central point in the MEDEIA Architecture is the Automation Component Implementation Model (ACIM). It consists of different models, which encapsulate all information, which needs to be specified to allow the implementation for the desired platform(s).



**Fig. 2.** Automation Component Implementation Model

### 3.2 Automation Component Model (ACM)

The Automation Component Model (ACM) contains implementation/platform independent aspects of ACs: behavior, diagnostics, interfaces, and the generic plant interface and plant behavior.

### 3.3 Execution System Model (ESM)

Within this model, the concrete hardware for an implementation is modeled. Information about processor, FPGA, memory, runtime environment, operating system, etc. is included in the System Model as well as loop points, which represent sensors and actuators.

### 3.4 Mapping Model (MM)

The gap between the platform independent ACM and the ESM is closed by the Mapping Model (MM). Generic plant interfaces and the concrete I/Os, provided by a specific system, are mapped as well as the functionalities provided by sub ACs.

### 3.5 Model Transformation

The transformation between the different models is the second key element within the MEDEIA approach. As can be seen in Fig. 2, there are different kind of models used in the MEDEIA framework. It is necessary to define appropriate transformations between the different models in order to supply the user with an automatic framework that provides exactly the information the user wants to specify/view/edit at the moment.

### 3.6 Integrated Diagnostics and Plant Specification

As mentioned before, the ACM includes also diagnostics and plant information along with the behavioral part. Based on diagnostics concepts and information about the controlled plant, appropriate DSVs have to be used to include the proper information in the MEDEIA approach.

### 3.7 Platform Oriented Merged Automation Component Model (POMAC)

Into this additional model all information, necessary for the implementation on specific platforms, is merged from the ACIM. The amount of information shall be reduced, increasing the efficiency of the code generation.

## 4 Service Robotics Domain

The domain of Service Robotics is one of the latest developments in the robotics area. It is connected with the trend that automation happens more and more outside of factory automation. The robots being used in this domain are called Service Robots. Their fields for automation cover:

- Laboratory automation (pharmaceutical, chemical, life science industries),
- Medical industry, rehabilitation and human care,
- Cleaning solutions,
- Security and homeland security, and
- Space and naval exploration.

In most cases these Service Robotics solutions need sophisticated specialized robots, tailored to the specific task. In order to react on changing environments and to save humans from harm, service robots are equipped with additional sensor feedback (e.g. visual, tactile, distance sensors).

The company Schunk [5] provides such special service robotic solutions based on very flexible mechatronical modules as shown in the following figures:



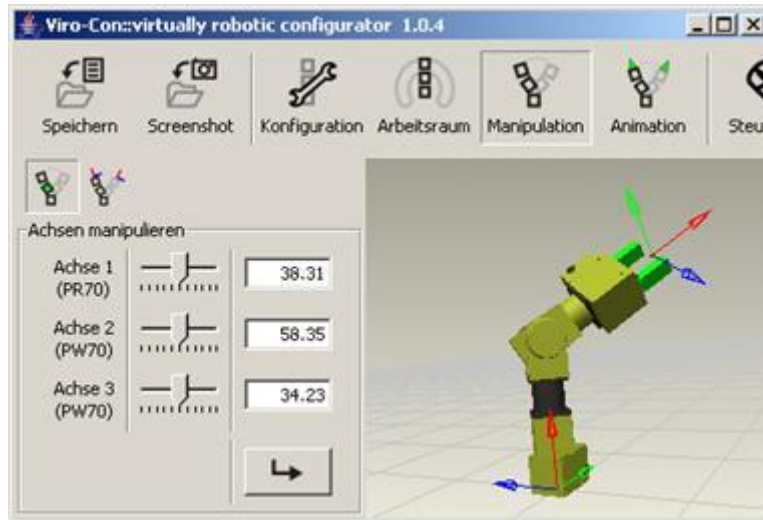
**Fig. 3.** PowerCube Module Overview – Standard Modules

#### 4.1 Demonstrator Description

The solution that is being provided is first being simulated in a virtual environment beforehand. This virtual environment is called Virtually Robot Configurator (Viro-Con). This is a software tool for configuration of robot systems, using the Schunk mechatronics PowerCube modules (e.g. rotary, lineary), providing the following functionality:

- Virtual assembly of robots (and environment obstacles) with 3D display
- Calculation and graphical representation of working range for any configuration
- Animation via direct manipulation and choose of coordinates
- Calculation of inverse kinematics
- Collision check (self-collision, obstacles)
- Output results in format of text files, pictures (3D area, jpg, bmp, png), videos

A part list is created by the tool, including actuators, connectors and user defined parameters. Tables are created by the tool for a trajectory (path points of tool center point positions and axes positions for animation), the structure describing parameters in DH-style (Denavit-Hartenberg) and tool center point positions in user defined coordination. A screenshot of the tool is depicted in Fig. 4.

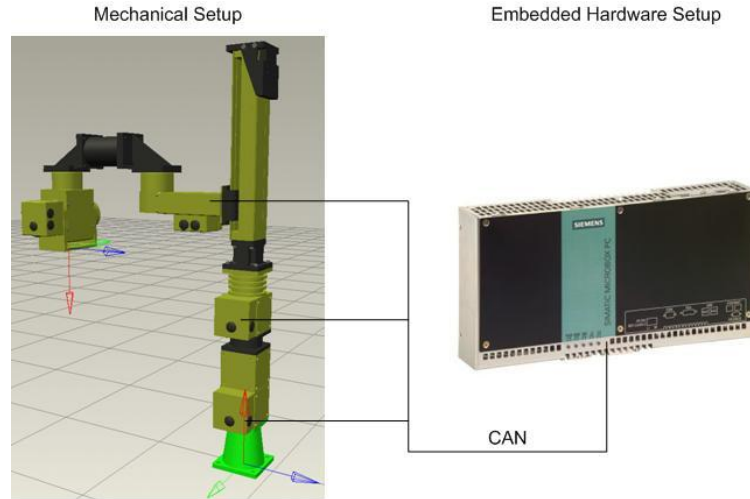


**Fig. 4.** Viro-Con Configuration 3D-Setup

Once a configuration has been designed in the virtual space, it can be manufactured and send to the customer. The goal of the demonstration is to cover the whole workflow from the simulation environment to the generation of control code for the supervisory controller.

The demonstrator for the use case will be a SCARA robot configuration, as depicted in Fig. 5. The SCARA robot consists out of different mechatronic PowerCube modules that are plugged together to form this service robot. The robot

will be controlled by one embedded controller from Siemens (i.e. Siemens MicroBox PC) which is equipped with the IEC 61499 control approach. Based on the motions defined in the Viro-Con tool, the control application for the real-life SCARA robot is generated and deployed to the Siemens controller.



**Fig. 5.** Schunk Demonstrator Embedded Hardware Setup

## 5 Conclusions and Future Work

The goal of MEDEIA is provide a pioneering methodology and a prototypical design and engineering framework for embedded systems design, which will enable the industrial automation industry to reduce design time and the costs for the development of complex control systems. Different meta-models are defined that enable the user to define the system in its own Domain Specific View. From there it will be transformed in various other internal models, which will at the end be transformed in executable code. A formal framework for embedded systems design will be prototyped, capable of managing the development of component-based design of embedded industrial control systems. The framework will include methods for modeling, simulation and verification of different industrial systems. One of the industrial use-cases in the MEDEIA project is the development of control code for modular service robotics, which is presented in this paper.

## 6 Acknowledgement

The work leading to this invention has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement No 211448. Further information is available at [www.medeia.eu](http://www.medeia.eu).

## 7 References

1. ARTIST2, Roadmap on Real-Time Techniques in Control System Implementation – Control for Embedded Systems Cluster, EU/IST FP6 Artist2 NoE, [www.artist-embedded.org](http://www.artist-embedded.org), (2006)
2. S. Kuppinger: Die Schlüssel zur Effizienz – Integration von Motion und Logic, IEE (Industrie elektr+elektronik), Hüthig Verlag, vol. 51, Nb. 1, (2006)
3. A. Hirzle, AutomationML<sup>TM</sup>, Press Conference, Hannover Messe – HMI, DaimlerChrysler AG, (2007)
4. T. Tømmila, et al., Next generation of industrial automation - Concepts and architecture of a component-based control system, VTT Research Notes 2303, (2005)
5. Schunk GmbH & Co. KG, "Powercube Mechatronic Components," Access Date July 2010. [Online]. Available: [www.powercube.de](http://www.powercube.de)